Desarrollo de un programa de computador para el análisis de estructuras en tres dimensiones tipo pórtico sometidas a cargas estáticas

Cristian Danilo Ramirez Vargas

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Civil y Agrícola
Bogotá D. C., Colombia
2020

Desarrollo de un programa de computador para el análisis de estructuras en tres dimensiones tipo pórtico sometidas a cargas estáticas

Cristian Danilo Ramirez Vargas

Tesis presentada como requisito parcial para optar al título de:
Magister en Estructuras

Director(a):
Ph. D. Martín Estrada Mejia

Línea de Investigación:
Análisis de estructuras
Grupo de Investigación:
Análisis, diseño y materiales - GIES

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Civil y Agrícola
Bogotá D. C., Colombia
2020

Hombres inteligentes gran pensantes
Hemos creado un monstruo
Las bombas radiactivas y nucleares
Que descompondrán la humanidad
Quien totalmente se autodestruirá

Ya creador no hay para volver a comenzar
Como dijo la sagrada maldición
El universo en siete días lo creo

Razas de todos los colores
Tomemos una reacción
Potencias monopolizadoras
Analicen esta situación
Países tercermundistas
De brazos no nos crucemos
Acabemos pronto con esto

Futuro nunca habrá
Futuro nunca ha habido
Este en mundo que esta perdido
Dependiendo de un botón
Y de la decisión
De un idealista cabrón

La tercera guerra mundial
Será un estruendo nuclear
Donde historiadores no podrán narrarla
Y los humanos no podremos resistirla

Las invenciones científicas
Lejos de liberar de la ignorancia
Y del trabajo envilecedor
Lo aumentan
Y hacen más refinada la servidumbre

—*La ciencia de la autodestrucción*, La Pestilencia (1989)

Agradecimientos

No habría podido hacer este trabajo sin la dirección del profesor Martín Estrada. Su conocimiento del mundo de la programación me ayudó en momentos decisivos durante el desarrollo del código. Gracias a él trabajé con la librería *three.js*. No sé como hacer para agradecerle por su paciencia.

Este trabajo también se debe al curso *Computación Visual* del profesor Jean Pierre Charalambos. Su descripción del *grafo* y como trabajar con la *escena* fue lo que me permitió hacer *FEM.js*.

Adicionalmente, apliqué el concepto de *cuaternión* en el problema de la rotación de los ejes de referencia tiempo después de haberlo estudiado en una de sus clases, lo que me permitió implementar el método de análisis matricial de manera innovadora. Gracias a su curso ahora creo entender muchas cosas que de adolescente siempre quise saber.

También quiero agradecer a la profesora Maritzabel Molina ya que mi entendimiento del método de análisis matricial proviene de su curso *análisis estructural básico*. A ella nos debemos muchos ingenieros estructurales.

Así mismo, quiero agradecer al profesor Fernando Ramírez, de la Universidad de los Andes, por enseñarme el *método de los elementos finitos*, y al profesor Dorian Linero por enseñarme a implementarlo. A ellos gracias por haberme permitido ganar confianza con el método.

Finalmente, quiero agradecer la ayuda de la Coordinación Curricular del Posgrado en Estructuras, especialmente a la profesora Caori Takeuchi quien no tuvo reparos en dejarme ver el curso Computación Visual. Ese día comenzó la verdadera tesis.

Resumen

El resumen es una presentación abreviada y precisa (la NTC 1486 de 2008 recomienda revisar la norma ISO 214 de 1976). Se debe usar una extensión máxima de 12 renglones. Se recomienda que este resumen sea analítico, es decir, que sea completo, con información cuantitativa y cualitativa, generalmente incluyendo los siguientes aspectos: objetivos, diseño, lugar y circunstancias, pacientes (u objetivo del estudio), intervención, mediciones y principales resultados, y conclusiones. Al final del resumen se deben usar palabras claves tomadas del texto (mínimo 3 y máximo 7 palabras), las cuales permiten la recuperación de la información.

Palabras clave: (máximo 10 palabras, preferiblemente seleccionadas de las listas internacionales que permitan el indizado cruzado).

A continuación se presentan algunos ejemplos de tesauros que se pueden consultar para asignar las palabras clave, según el área temática:

Artes: AAT: Art y Architecture Thesaurus.
Ciencias agropecuarias: 1) Agrovoc: Multilingual Agricultural Thesaurus - F.A.O. y 2)GEMET: General Multilingual Environmental Thesaurus.
Ciencias sociales y humanas: 1) Tesauro de la UNESCO y 2) Population Multilingual Thesaurus.
Ciencia y tecnología: 1) Astronomy Thesaurus Index. 2) Life Sciences Thesaurus, 3) Subject Vocabulary, Chemical Abstracts Service y 4) InterWATER: Tesauro de IRC - Centro Internacional de Agua Potable y Saneamiento.
Tecnologías y ciencias médicas: 1) MeSH: Medical Subject Headings (National Library of Medicine's USA) y 2) DECS: Descriptores en ciencias de la Salud (Biblioteca Regional de Medicina BIREME-OPS).
Multidisciplinarias: 1) LEMB - Listas de Encabezamientos de Materia y 2) LCSH- Library of Congress Subject Headings.

También se pueden encontrar listas de temas y palabras claves, consultando las distintas bases de datos disponibles a través del Portal del Sistema Nacional de Bibliotecas[1], en la sección Recursos bibliográficos.ºpción "Bases de datos".

Abstract

Es el mismo resumen pero traducido al inglés. Se debe usar una extensión máxima de 12 renglones. Al final del Abstract se deben traducir las anteriores palabras claves tomadas del

[1] ver: www.sinab.unal.edu.co

texto (mínimo 3 y máximo 7 palabras), llamadas keywords. Es posible incluir el resumen en otro idioma diferente al español o al inglés, si se considera como importante dentro del tema tratado en la investigación, por ejemplo: un trabajo dedicado a problemas lingüísticos del mandarín seguramente estaría mejor con un resumen en mandarín.

Keywords: palabras clave en inglés(máximo 10 palabras, preferiblemente seleccionadas de las listas internacionales que permitan el indizado cruzado)

Contenido

Lista de Figuras

Lista de Tablas

1 Ejemplos de aplicación

Con el fin de presentar el funcionamiento de los programas desarrollados, en este capítulo se presentan diferentes modelos de estructuras tridimensionales de elementos rectos sometidos a cargas estáticas, los cuales se resuelven usando la librería *pyFEM* o el programa de computador *StressUN*.

1.1. Funcionamiento de la librería pyFEM

Se presentan varios ejemplos provenientes de los libros de la referencia bibliográfica, los cuales se resuelven usando el programa *StressUN* o la librería *pyFEM*, con el objetivo de comparar la respuesta obtenida con la presentada en dichos textos.

1.1.1. Cercha plana

En el libro *Microcomputadores en Ingeniería Estructural*, del ingeniero *Jairo Uribe Escamilla* (Escamilla, 1995), se presenta el ejemplo y la solución del modelo de una cercha simple plana estáticamente determinada (una cercha simple plana es una estructura compleja generada a partir de una estructura triangular base **beer1997mecanica**). El ejercicio consiste en encontrar los desplazamientos de los nudos, las fuerzas de las reacciones y las fuerzas de cada elemento de la cercha mostrada en la figura **??**. La solución presenta las matrices de rigidez de cada uno de los elementos de la cercha, la matriz de rigidez de toda la estructura, la solución de los desplazamientos de los nodos, las reacciones de los apoyos y las fuerzas internas de cada elemento.

A continuación se presenta la solución de dicho ejemplo, el cual se resuelve usando la librería `pyFEM`.

Ejemplo 7.1 - *Resuelva completamente la cercha mostrada por el método matricial de los desplazamientos. El material es acero estructural con* $E = 2040\ t/cm^2$. *Las áreas están dadas entre paréntesis en* cm^2.

Solución - Para solucionar el ejemplo anterior usando la librería `pyFEM`, se ingresa la información necesaria para describir completamente el modelo de la estructura mediante ins-

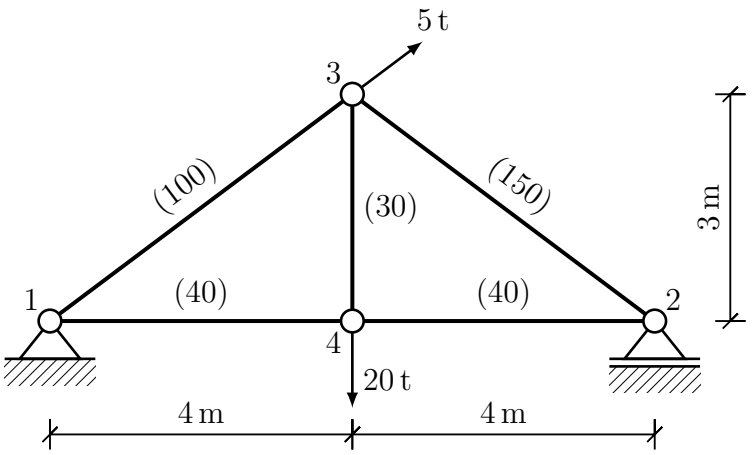

Figura 1-1: Cercha simple plana del *Ejemplo 7.1* de Escamilla, 1995.

trucciones en el lenguaje de programación `python`, las cuales se almacenan en un archivo de texto plano. Dicha información consiste en: (1) los materiales, (2) las secciones transversales, (3) los nodos, (4) los elementos tipo cercha, (5) los apoyos y (6) las cargas.

En el algoritmo **??** se presentan las instrucciones que debe recibir `pyFEM` para solucionar el modelo de la estructura. Las instrucciones consisten en crear un nuevo objeto tipo `Structure`, al cual se le ha dado el nombre de *structure* y, seguidamente, se agregan los materiales, las secciones, los nodos, los elementos tipo cercha, los apoyos, los patrones de carga y las cargas en los nodos. Una vez los datos del modelo de la estructura han sido ingresados, se puede solucionar el modelo mediante la instrucción `structure.solve()`.

Algoritmo 1.1: Ingreso de los datos del modelo de la estructura a *pyFEM*.

```
# create structure
structure = Structure()

# add material
structure.materials.add("acero", 2040e4)

# add sections
structure.sections.add("section1", "acero", 30e-4)
structure.sections.add("section2", "acero", 40e-4)
structure.sections.add("section3", "acero", 100e-4)
structure.sections.add("section4", "acero", 150e-4)

# add nodes
structure.nodes.add('1', 0, 0, 0)
structure.nodes.add('2', 8, 0, 0)
structure.nodes.add('3', 4, 3, 0)
```

```
structure.nodes.add('4', 4, 0, 0)

# add trusses
structure.trusses.add('1-3', '1', '3', "section3")
structure.trusses.add('1-4', '1', '4', "section2")
structure.trusses.add('3-2', '3', '2', "section4")
structure.trusses.add('4-2', '4', '2', "section2")
structure.trusses.add('4-3', '4', '3', "section1")

# add support
structure.supports.add('1', True, True, True)
structure.supports.add('2', False, True, True)
structure.supports.add('3', False, False, True)
structure.supports.add('4', False, False, True)

# add load pattern
structure.load_patterns.add("point loads")

# add point loads
structure.load_patterns["point loads"].point_loads.add('4', 0, -20, 0)
structure.load_patterns["point loads"].point_loads.add('3', 5*0.8, 5*0.6, 0)

# solve the structure
structure().solve()
```

Cuando se ejecuta la instrucción `structure.solve()`, `pyFEM` comienza a solucionar el modelo de la estructura con base en la información ingresada. Los pasos que se efectúan para solucionar la estructura consisten en: (1) asignar los grados de libertad de los nodos, (2) ensamblar la matriz de rigidez del modelo de la estructura, (3) imponer las condiciones de apoyo en la matriz de rigidez del modelo, (4) ensamblar el vector de fuerzas en los nodos para cada uno de los patrones de carga, (5) imponer las condiciones de apoyo en el vector de fuerzas en los nodos para cada caso de carga, (6) encontrar los desplazamientos de los nodos para cada patrón de carga, (7) encontrar las reacciones en los apoyos para cada patrón de carga y (8) guardar la solución en los nodos y en los apoyos para cada patrón de carga.

A continuación se presenta el resultado de cada uno de los pasos realizados por `pyFEM` para solucionar la cercha del ejemplo 7.1.

Grados de libertad de los nodos

Para realizar el ensamblaje de la matriz de rigidez del modelo de la estructura y del vector de fuerzas de los nodos, se asignan números a los grados de libertad de los nodos de la estructura en el orden en que fueron ingresados.

Con base en ésto y en el algoritmo **??**, el nodo *'1'* se le han asignado los grados de libertad *0, 1* y *2*, al nodo *'2'* los grados de libertad *3, 4* y *5*, y así sucesivamente.

Matrices de rígidez

Una vez se establecen los grados de libertad de los nodos del modelo de la estructura, se ensambla la matriz de rigidez del modelo de la estructura. Este proceso consiste en calcular una a una las matrices de rigidez de los elementos ensamblandolas en la matriz de rigidez del modelo, la cual, inicialmente, es una matriz de ceros.

Aunque no se almacenan las matrices de rigidez de cada uno de los elementos del modelo de la estructura, el usuario puede consultarlas. En la tabla **??** se presenta la matriz de rigidez en coordenadas globales del elemento *1-3*, con sus respectivos grados de libertad, la cual se obtiene mediante la instrucción *structure.trusses['1-3'].get_global_stiff_matrix()*.

	0	*1*	*2*	*6*	*7*	*8*
0	26112	19584	0	-261112	-19584	0
1	19584	14688	0	-19584	-14688	0
2	0	0	0	0	0	0
3	-26112	-19584	0	26112	19584	0
4	-19584	-14688	0	19584	14688	0
5	0	0	0	0	0	0

t/m

Tabla 1-1: Matriz de rigidez en coordenadas globales del elemento *1-3*.

En las tablas **??** a la **??** se presentan las matrices de rigidez en coordenadas globales de los demás elementos de la estructura, las cuales se obtienen con instrucciones similares a la anterior.

	0	*1*	*2*	*9*	*10*	*11*
0	20400	0	0	-20400	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
9	-20400	0	0	20400	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0

t/m

Tabla 1-2: Matriz de rigidez en coordenadas globales del elemento *1-4*.

	6	*7*	*8*	*3*	*4*	*5*	
6	39168	-29376	0	-39168	29376	0	
7	-29376	22032	0	29376	-22032	0	
8	0	0	0	0	0	0	
3	-39168	29376	0	39168	-29376	0	t/m
4	29376	-22032	0	-29376	22032	0	
5	0	0	0	0	0	0	

Tabla 1-3: Matriz de rigidez en coordenadas globales del elemento *3-2*.

	9	*10*	*11*	*3*	*4*	*5*	
9	20400	0	0	-20400	0	0	
10	0	0	0	0	0	0	
11	0	0	0	0	0	0	
3	-20400	0	0	20400	0	0	t/m
4	0	0	0	0	0	0	
5	0	0	0	0	0	0	

Tabla 1-4: Matriz de rigidez en coordenadas globales del elemento *4-2*.

	9	*10*	*11*	*6*	*7*	*8*	
9	0	0	0	0	0	0	
10	0	20400	0	0	-20400	0	
11	0	0	0	0	0	0	
6	0	0	0	0	0	0	t/m
7	0	-20400	0	0	20400	0	
8	0	0	0	0	0	0	

Tabla 1-5: Matriz de rigidez en coordenadas globales del elemento *4-3*.

Al comparar los resultados obtenidos con las matrices de rigidez de cada uno de los elementos presentados en Escamilla, 1995 se encuentra que se trata de los mismos valores.

Así mismo como el usuario puede indagar por la matriz de rigidez en coordenadas globales de cada elemento, puede hacerlo para el modelo de la estructura, mediante la instrucción *structure.get_k()*. Al hacerlo, obtiene los valores presentados en la tabla **??**.

Una vez más, al comparar los resultados obtenidos con la matriz de rigidez del modelo de la estructura presentado en Escamilla, 1995 se encuentra que se trata de los mismos valores.

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
0	46512	19584	0	0	0	0	-26112	-19584	0	-20400	0	0
1	19584	14688	0	0	0	0	-19584	-14688	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	59568	-29376	0	-39168	29376	0	-20400	0	0
4	0	0	0	-29376	22032	0	29376	-22032	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	-26112	-19584	0	-39168	29376	0	65280	-9792	0	0	0	0
7	-19584	-14688	0	29376	-22032	0	-9792	57120	0	0	-20400	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	-20400	0	0	-20400	0	0	0	0	0	40800	0	0
10	0	0	0	0	0	0	0	-20400	0	0	20400	0
11	0	0	0	0	0	0	0	0	0	0	0	0

t/m

Tabla 1-6: Matriz de rigidez en coordenadas globales del modelo de la estructura.

Sin embargo, se debe reordenar una de las dos matrices para encontrar los valores en las mismas posiciones de la matriz.

Obtenida la matriz de rigidez de la estructura, se procede a imponer las condiciones de apoyo del modelo de la estructura. Esto se realiza modificando la estructura, tal como se mostró en el capítulo **??**. En la tabla **??** se presenta dicho resultado.

Vector de fuerzas

Así como se deben encontrar las matrices de rigidez de cada uno de los elementos del modelo de la estructura para posteriormente *ensamblarlas*, se deben encontrar las acciones en los nodos de los elementos para cada patrón de carga.

El usuario puede indagar por dicho vector, para el patrón de carga *point loads* mediante la instrucción `structure.load_patterns['point loads'].get_f()`. Con dicha instrucción, el usuario obtiene los datos que se muestran en la tabla **??**.

Obtenido el vector de fuerzas para dicho patrón de carga, se imponen las condiciones de apoyo del modelo de la estructura. Debido a que los desplazamiento en los apoyos son iguales a cero, el vector de fuerzas en los nodos no varia.

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	
0	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	0	0	0	0	
2	0	0	1	0	0	0	0	0	0	0	0	0	
3	0	0	0	59568	0	0	-39168	29376	0	-20400	0	0	
4	0	0	0	0	1	0	0	0	0	0	0	0	
5	0	0	0	0	0	1	0	0	0	0	0	0	
6	0	0	0	-39168	0	0	65280	-9792	0	0	0	0	
7	0	0	0	29376	0	0	-9792	57120	0	0	-20400	0	
8	0	0	0	0	0	0	0	0	1	0	0	0	
9	0	0	0	-20400	0	0	0	0	0	40800	0	0	
10	0	0	0	0	0	0	0	-20400	0	0	20400	0	
11	0	0	0	0	0	0	0	0	0	0	0	1	t/m

Tabla 1-7: Matriz de rigidez en coordenadas globales del modelo de la estructura con las condiciones de apoyo impuestas.

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	
0	0	0	0	0	0	4	3	0	0	-20	0	^{t} t

Tabla 1-8: Vector de fuerzas de los nodos del modelo de la estructura para el patrón de carga *point loads*.

Vector de desplazamientos

Al imponerse las condiciones de apoyo a la matriz de rigidez del modelo de la estructura, se determinan los desplazamientos de los nodos de la estructura para cada uno de los patrones de carga, donde se ha calculado el vector de fueras en los nodos de cada patrón de cargas y se le han impuesto las condiciones de apoyo.

En la tabla **??** se presenta el vector de desplazamientos en los nodos del modelo de la estructura para el patrón de carga *point loads*, los cuales son iguales a los presentados en Escamilla, 1995.

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	
0	0	0	1.307	0	0	0.645	-1.337	0	0.654	-2.317	0	^{t} 1×10^{-3} m

Tabla 1-9: Vector de desplazamientos de los nodos del modelo de la estructura para el patrón de carga *point loads*.

Vector de reacciones

Una vez se encuentran los desplazamientos en los nodos del modelo de la estructura, para cada uno de los patrones de carga, se procede a encontrar las fuerzas en los nodos producto de dichos desplazamientos.

En la tabla **??** se presenta el vector de fuerzas en los nodos del modelo de la estructura para el patrón de carga *point loads*, los cuales son iguales a los presentados en Escamilla, 1995.

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	
{-4	7	0	0	10	0	4	3	0	0	-20	0 }	^{t} t

Tabla 1-10: Vector de fuerzas de los nodos del modelo de la estructura para el patrón de carga *point loads*.

Procesamiento de los resultados

Hasta aquí, se ha solucionado el modelo de la estructura sometida al patrón de carga *point loads*. Con los resultados almacenados, se pueden determinar otros resultados que son interesantes. Para el caso concreto del modelo objeto de estudio en esta sección, se puede querer determinar las fuerzas internas de los elementos o las defomaciones en los mismos.

A modo de ejemplo, en la tabla **??** se presenta las fuerzas internas de cada uno de los elementos del modelo de la estructura sometidos al patrón de carga *point loads*, los cuales son prácticamente iguales a los presentados en Escamilla, 1995.

Elemento	Fuerza [t]
1-3	-11.667
1-4	13.333
3-2	-16.667
4-2	13.333
4-3	20

Tabla 1-11: Fuerzas internas de los elementos del modelo de la estructura para el patrón de carga *point loads*.

1.1.2. Pórtico tridimensional

En el libro *Microcomputadores en Ingeniería Estructural*, del ingeniero *Jairo Uribe Escamilla* (Escamilla, 1995), se presenta el ejemplo y la solución del modelo de un pórtico tridimensio-

nal. A continuación se presenta la solución de dicho ejemplo, el cual se resuelve usando la librería `pyFEM`.

Ejemplo 7.6 - *Resuelva matricialmente el pórtico de la figura.*

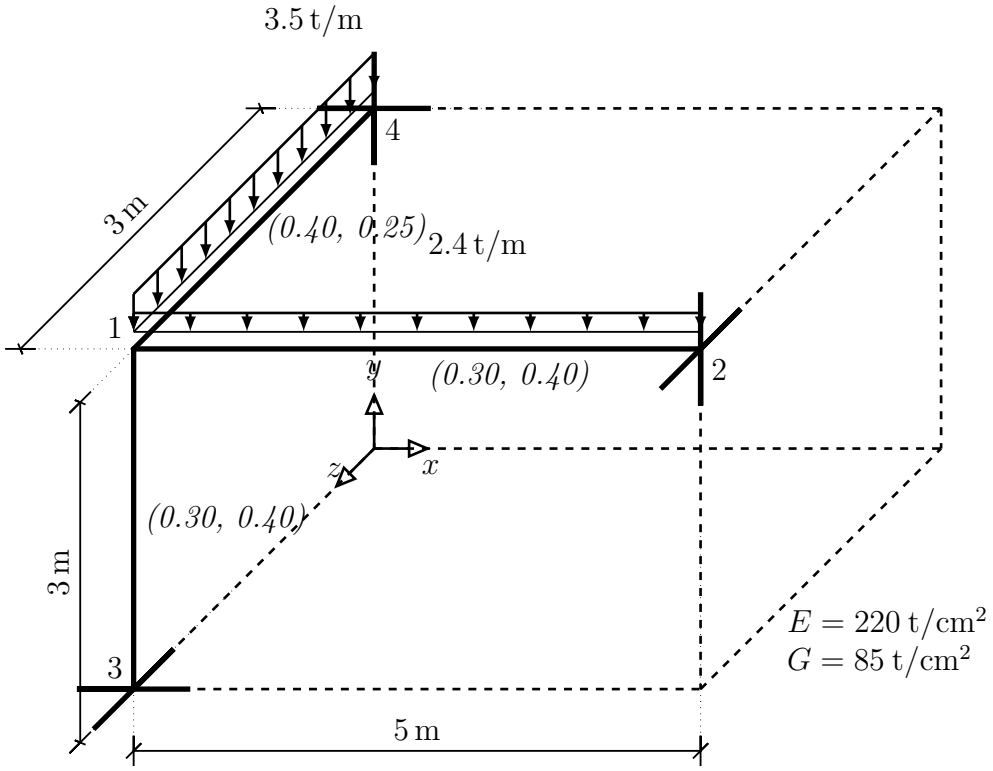

Figura 1-2: Pórtico tridimensional del *Ejemplo 7.6* de Escamilla, 1995.

Solución - La figura a la cual hace referencia el ejemplo 7.6 se presenta en la figura **??**. En ésta se presenta la geometría del modelo de la estructura, así como las propiedades del material, las cargas a las que están sometidos y las condiciones de apoyo.

Para solucionar el ejemplo anterior usando la librería `pyFEM`, se ingresa la información necesaria para describir completamente el modelo de la estructura mediante instrucciones en el lenguaje de programación `python`, las cuales se almacenan en un archivo de texto plano. Dicha información consiste en: (1) los materiales, (2) las secciones transversales, (3) los nodos, (4) los elementos tipo pórtico, (5) los apoyos y (6) las cargas.
En el algoritmo **??** se presenta las instrucciones que debe recibir `pyFEM` para solucionar el modelo de la estructura. Las instrucciones consisten en crear un nuevo objeto tipo `Structure`, al cual se le ha dado el nombre de *structure* y, seguidamente, se agregan los materiales, las secciones, los nodos, los elementos tipo pórtico, los apoyos, los patrones de carga y las cargas

en los elementos. Una vez los datos del modelo de la estructura han sido ingresado, se puede solucionar el modelo mediante la instrucción `structure.solve()`.

Algoritmo 1.2: Ingreso de los datos del modelo de la estructura a `pyFEM`.

```
# structure
structure = Structure()

# add material
structure.materials.add('material1', 220e4, 85e4)

# add sections
structure.sections.add('section1', 'material1', 0.12, 9e-4, 1.6e-3, 1.944e-3)
structure.sections.add('section2', 'material1', 0.10, 1.333e-3, 5.208e-4,
    1.2734e-3)

# add nodes
structure.nodes.add('1', 0, 3, 3)
structure.nodes.add('2', 5, 3, 3)
structure.nodes.add('3', 0, 0, 3)
structure.nodes.add('4', 0, 3, 0)

# add frames
structure.frames.add('1-2', '1', '2', 'section1')
structure.frames.add('3-1', '3', '1', 'section1')
structure.frames.add('4-1', '4', '1', 'section2')

# add supports
structure.supports.add('2', *6 * (True,))
structure.supports.add('3', *6 * (True,))
structure.supports.add('4', *6 * (True,))

# add load pattern
structure.load_patterns.add("distributed loads")

# add distributed loads
structure.load_patterns["distributed loads"].distributed_loads.add('1-2', 0,
    -2.4, 0)
structure.load_patterns["distributed loads"].distributed_loads.add('4-1', 0,
    -3.5, 0)

# solve
structure.solve()
```

Cuando se ejecuta la instrucción `structure.solve()`, `pyFEM` comienza a solucionar el modelo de la estructura con base en la información ingresada. A continuación se realiza la comparación entre los resultados obtenidos contra los resultados presentados en Escamilla, 1995.

Desplazamientos de los nodos

En la tabla **??** se presenta el desplazamientos de los nodos del modelo de la estructura para el patrón de carga *distributed loads*, los cuales son iguales a los presentados en Escamilla, 1995.

Nodo	1	2	3	4
u_x [m]	2.687×10^{-5}	0	0	0
u_y [m]	-1.157×10^{-4}	0	0	0
u_z [m]	-1.001×10^{-5}	0	0	0
r_x [rad]	-5.669×10^{-4}	0	0	0
r_y [rad]	7.905×10^{-6}	0	0	0
r_z [rad]	-6.309×10^{-4}	0	0	0

Tabla 1-12: Desplazamientos en los nodos del modelo para el patrón de carga *distributed loads*.

Reacciones de los apoyos

En la tabla **??** se presenta las reacciones de los apoyos del modelo de la estructura para el patrón de carga *distributed loads*, las cuales son practicamente iguales a las presentadas en Escamilla, 1995.

Nodo	2	3	4
F_x [t]	-1.419	1.438	-0.02
F_y [t]	6.572	1.019×10^{1}	5.742
F_z [t]	5.658×10^{-3}	-7.394×10^{-1}	0.734
T [t m]	1.873×10^{-1}	-7.350×10^{-1}	-3.146
M_y [t m]	1.102×10^{-2}	-4.354×10^{-3}	-0.037
M_z [t m]	-5.986	-1.417	0.228

Tabla 1-13: Reacciones de los apoyos del modelo para el patrón de carga *distributed loads*.

Bibliografía

Chacon, S. (2014). *Pro Git*. Berkeley, CA New York, NY, Apress,Distributed to the Book trade worldwide by Spring Science+Business Media.

Computers & Structures. (2017). *CSi Anlysis Reference Manual*.

Computers & Structures. (2019). *Welcome to ETABS*.

Computers & Structures. (2020). ETABS System Requirements [Accedido: 2020-09-29].

Escamilla, J. (1995). *Microcomputadores en ingeniería estructural*. Santafé de Bogotá, ECOE Universidad Nacional de Colombia. Facultad de Ingeniera.

Lutz, M. (2013). *Learning Python*. Sebastopol, CA, O'Reilly.

Wilson, E. L. & Dovey, H. H. (1972). Three dimensional analysis of building systems - TABS. *Earthwuake engineering research center*.

Wilson, E. L., Hollings, J. P. & Dovey, H. (1975). Three dimensional analysis of building systems (extended version). *Earthwuake engineering research center*.